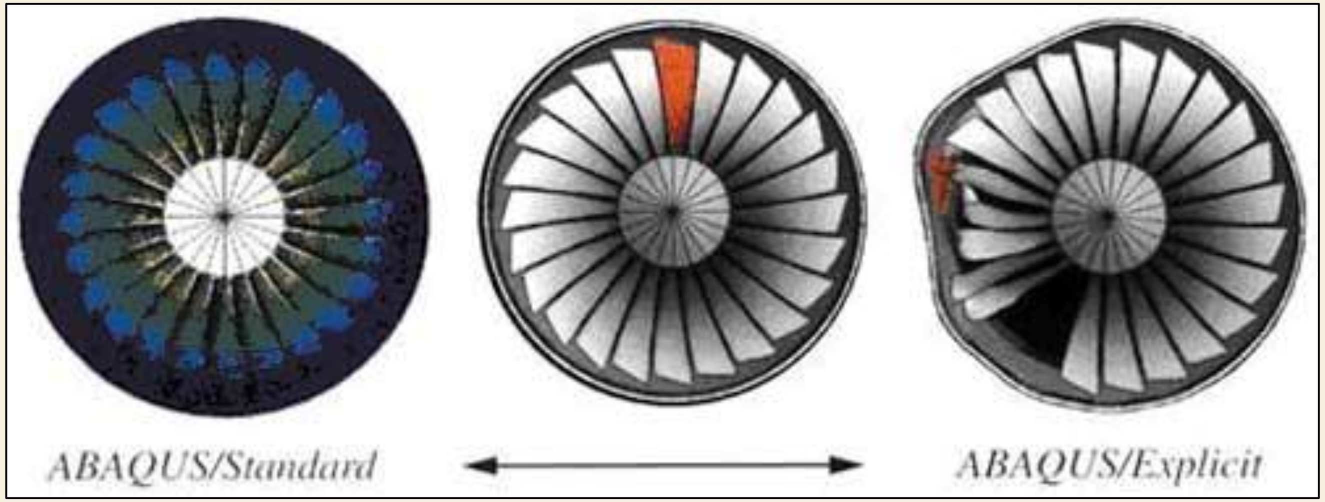
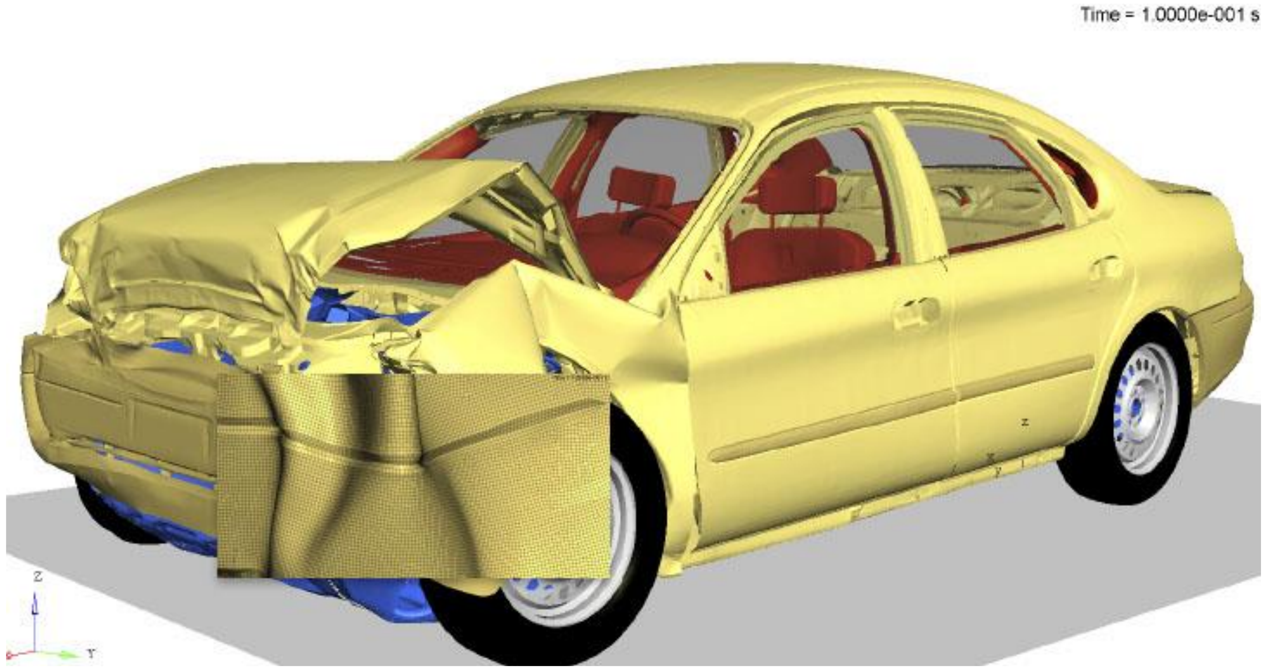
ЭЛЕМЕНТЫ ТЕХНИЧЕСКОГО ЗРЕНИЯ ПРТС

ОБЩИЕ ПОНЯТИЯ

ОСОБЕННОСТИ РЕШЕНИЯ ИНЖЕНЕРНЫХ ЗАДАЧ

- Практическая направленность
  - Конкретный результат





ОСОБЕННОСТИ РЕШЕНИЯ ИНЖЕНЕРНЫХ ЗАДАЧ

Значительный объём вычислений

Метод конечных элементов

Задачи обработки сигналов

СЛОЖНЫЕ МАТЕМАТИЧЕСКИЕ МОДЕЛИ

- Численные методы
- Необходимость сглаживания и фильтрации
- Спектральные характеристики
- …

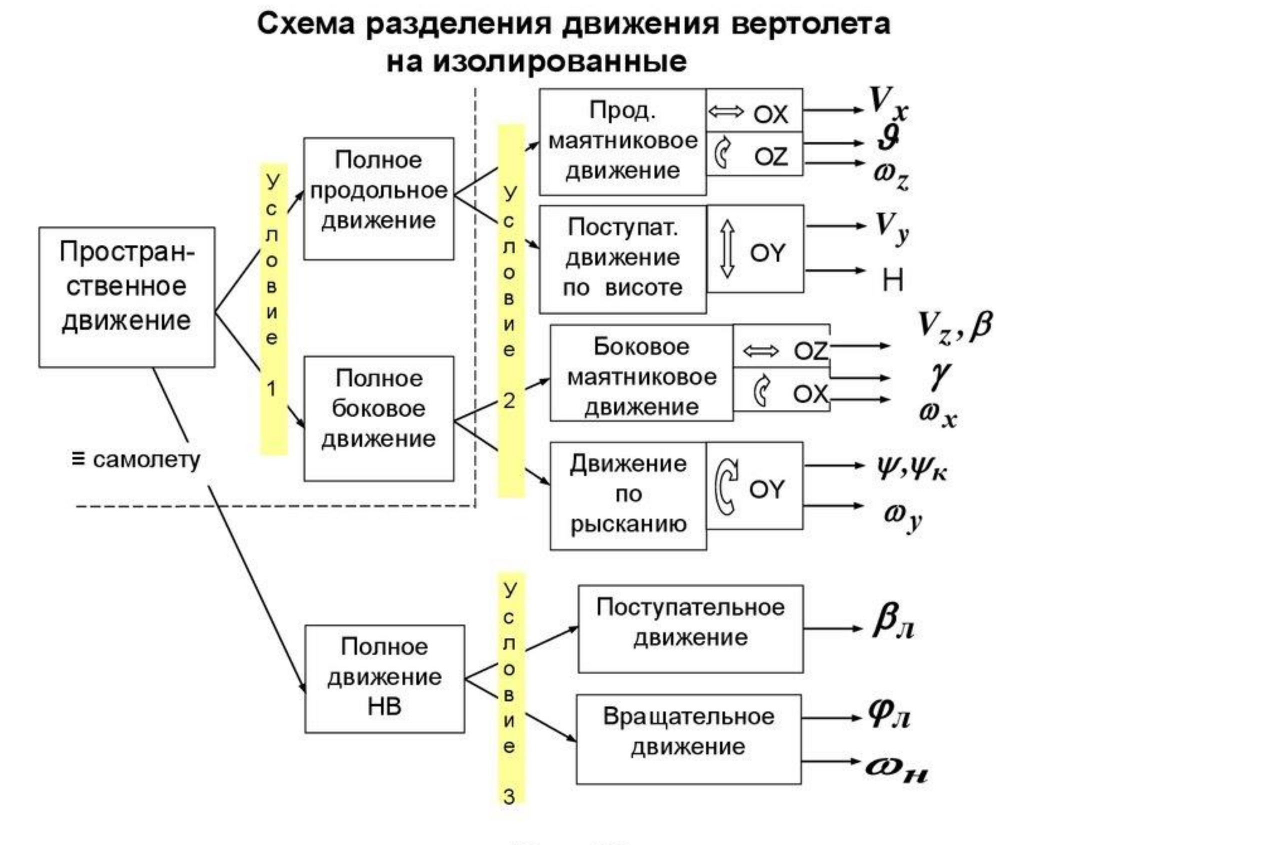


Рис. 4.3.

РЕШАЮТСЯ ИНЖЕНЕРАМИ

ЭТАПЫ РЕШЕНИЯ ЗАДАЧИ

Постановка
- Выбор модели
- Постановка вычислительной задачи

Анализ
- Выбор численного метода

Программирование

Отладка

Счёт

Обработка результата

Коррекция модели

ПОГРЕШНОСТЬ В РЕШЕНИИ

- Получить точное решение невозможно

$$u(x,\alpha) = y + \delta y$$

- У погрешности – несколько составляющих

$$\delta y = \delta_{\text{н}} y + \delta_{\text{м}}\, y + \delta_{\text{в}} y$$

ПОГРЕШНОСТЬ ВЫЧИСЛЕНИЙ

- Ограничения связаны с дискретизацией
- Разберём на примере десятеричной системы счисления:
  - 6 разрядов после запятой

$$\left.\begin{matrix} x_1 = 7.235673456 \\ x_2 = 7.235672954 \end{matrix}\right\} => x^* = 7.235673$$

  - Абсолютная погрешность

$$\Delta x = |x \ - x^*|$$

  - Относительная погрешность

$$\delta x = \frac{\Delta x}{|x|} = \frac{\Delta|x - x^*|}{|x|}$$

ПОГРЕШНОСТЬ ВЫЧИСЛЕНИЙ

- Погрешность операций:
  - Сложение $c = a + b$

$$c^* = a^* + b^* = a + b + \Delta a + \Delta b$$
$$\Delta c = c^* - c = \Delta a + \Delta b$$
$$\delta c = \frac{\Delta c}{|c|} = \frac{|\Delta \mathrm{a}| + |\Delta b|}{|a + b|}$$

  - Вычитание $c = a - b$

$$c^* = a^* - b^* = a - b + \Delta a - \Delta b$$
$$\Delta c = c^* - c = |\Delta a| + |\Delta b|$$
$$\delta c = \frac{\Delta c}{|c|} = \frac{|\Delta \mathrm{a}| + |\Delta b|}{|a - b|}$$

ПОГРЕШНОСТЬ ВЫЧИСЛЕНИЙ

- Погрешность операций
  - Умножение $c = a \cdot b$

$$c^* = a^* \cdot b^* = a \cdot b + a \cdot \Delta b + b \cdot \Delta a + \Delta a \cdot \Delta b$$
$$\Delta c = a\Delta b + b\Delta a \leq \Delta(a, b) \cdot \max(a, b)$$
$$\delta c = \frac{\max(\Delta a, \Delta b)}{\min(a, b)} \approx \max(\delta a, \delta b)$$

  - Деление $c = \frac{a}{b}$ – аналогично

$$\Delta c \approx \Delta(a, b) \cdot \max(a, \frac{1}{b})$$
$$\delta c = \frac{\max(\Delta a, \Delta b)}{\min(a, b)} \approx \max(\delta a, \delta\left(\frac{1}{b}\right))$$

ПОГРЕШНОСТЬ ВЫЧИСЛЕНИЙ

- В реальных задачах:
  - Переменные двоичные
  - Дробные числа – с плавающей запятой
  - Возникает проблема переполнения

- Например, double – 15 знаков мантиссы:
- A=3.141592653589793 // π
- B=0.7853981633974483 // $\frac{\pi}{4}$
- C=3.926990816987241 // A+B

ПОГРЕШНОСТЬ ВЫЧИСЛЕНИЙ

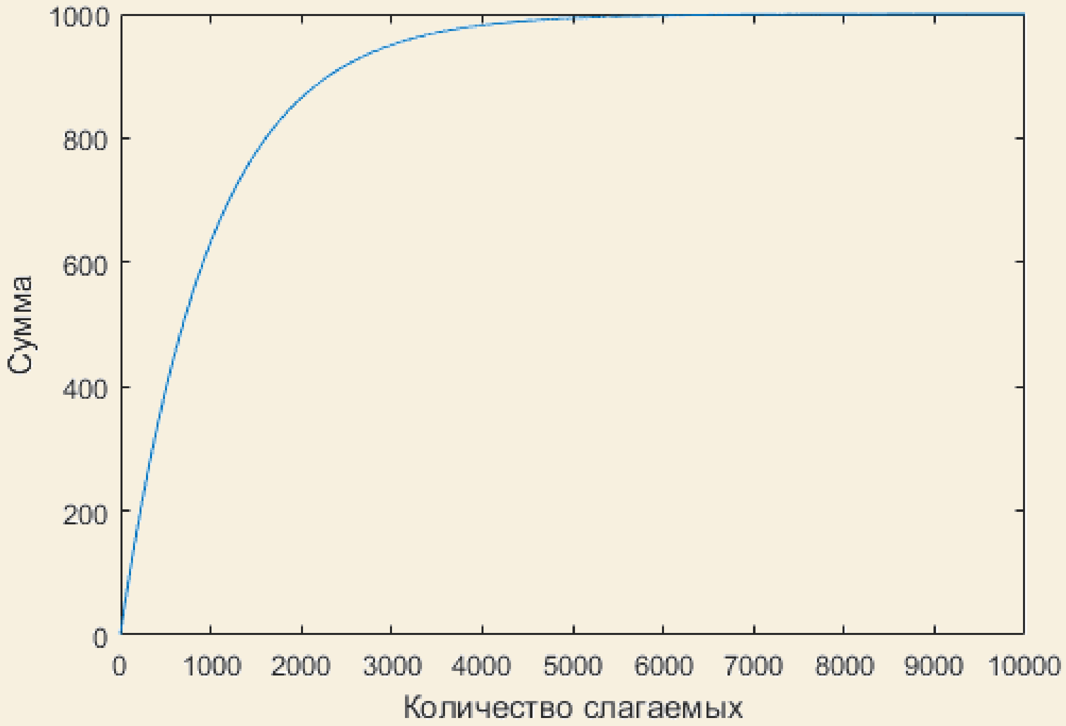
Ещё один (надуманный) пример

```
#include <iostream>
#define USE_MATH_DEFINES
#include <cmath>
#include <limits>

using namespace std;

int main(int argc, char *argv[])
{
		float summ1 = 0, summ2 = 0;
		double q = 0.999;
		int maxStep = 100000; // Просто большое число
		for (int i=0; i<= maxStep; i++)
		{
				summ1 += pow(q, i);
		}
		cout.precision(std::numeric_limits<float>::digits10);
		cout << "Summ1 = " << summ1 << " should be " << 1.0/(1-q) << endl;
		for (int i=maxStep; i>=0; i--)
		{
				summ2 += pow(q, i);
		}
		cout << "Summ2 = " << summ2 << " should be " << 1.0/(1-q) << endl;
		return 0;
}
```

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ



- Мы рассматриваем геометрическую прогрессию с основанием 0.999
- Точность получившейся суммы, в теории, определяется количеством слагаемых
- Суммы одинаковы, но отличаются порядком суммирования

- Summ1 = 999.978 and should be 1000
- Summ2 = 1000 and should be 1000

КОРРЕКТНОСТЬ ВЫЧИСЛИТЕЛЬНОЙ ЗАДАЧИ

X – множество допустимых входных данных

Y – множество возможных решений

Вычислительная задача корректна, если

1. Решение $y \in Y$ существует при $x \in X$
2. Решение единственно
3. Решение устойчиво по возмущениям X

КОРРЕКТНОСТЬ - СУЩЕСТВОВАНИЕ

- Естественное ограничение
- Нет решения:
  - Неправильная модель
  - Неправильная постановка задачи

$$x^2 + b \cdot x + c = 0$$

Решения $x \in D$ существуют только при

$$b^2 - 4ac \geq 0$$

КОРРЕКТНОСТЬ - ЕДИНСТВЕННОСТЬ

- Для математической задачи решение может быть неединственым
- Для вычислительной – вводятся дополнительные ограничения
  – Например, для квадратного уравнение – два корня

КОРРЕКТНОСТЬ - УСТОЙЧИВОСТЬ

- Непрерывная зависимость по входным данным

$$\forall \varepsilon > 0 \; \exists \delta(\varepsilon) > 0 \colon \forall x^* \colon \Delta(x^*) < \delta \Rightarrow \Delta(y^*) < \varepsilon$$

- В этом случае можно получить достаточно точное решение, увеличивая точность входных данных
- Это требование часто не выполняется!!!

ОБУСЛОВЛЕННОСТЬ ЗАДАЧИ

- Очень похожа на требование устойчивости:

$$\Delta(y^*) \leq M\Delta(x^*)$$

- M – число обусловленности

$$(y-1)^4 = 0 \;\; => \;\; y = 1$$

Внесём погрешность:

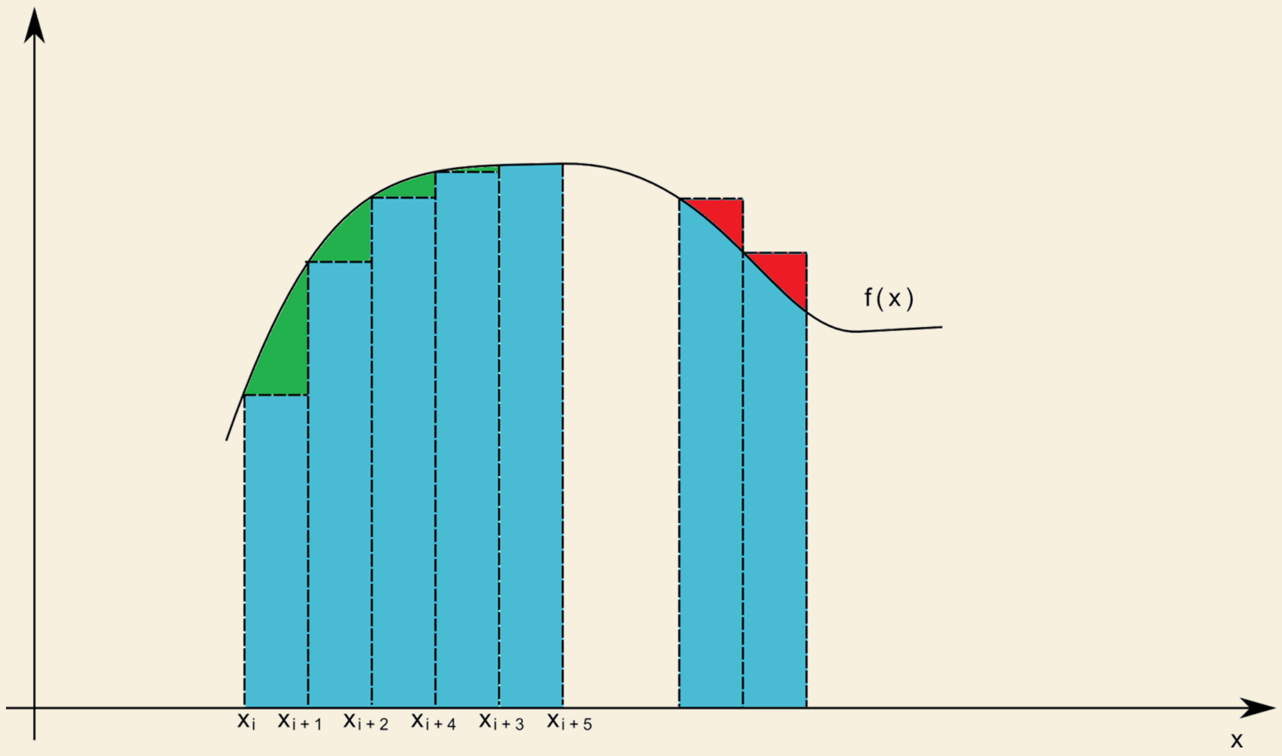
$$(y-1)^4 = 10^{-8} =>$$
$$y = (1 \pm 0.01, 1 \pm 0.01i)$$
$$\Delta\mathrm{x} = 10^{-8}, \Delta y = 10^{-2}$$

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ

- Классы методов:
  - Эквивалентных преобразований
  - Аппроксимации
  - Прямые (точные)
  - Итерационные
  - Статистических испытаний

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ

- Методы аппроксимации
  - Решение задачи, решение которой близко к исходной
  - Погрешность аппроксимации
  - Вообще говоря, не всегда сходится
  - Например, вычисление интегралов



ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ

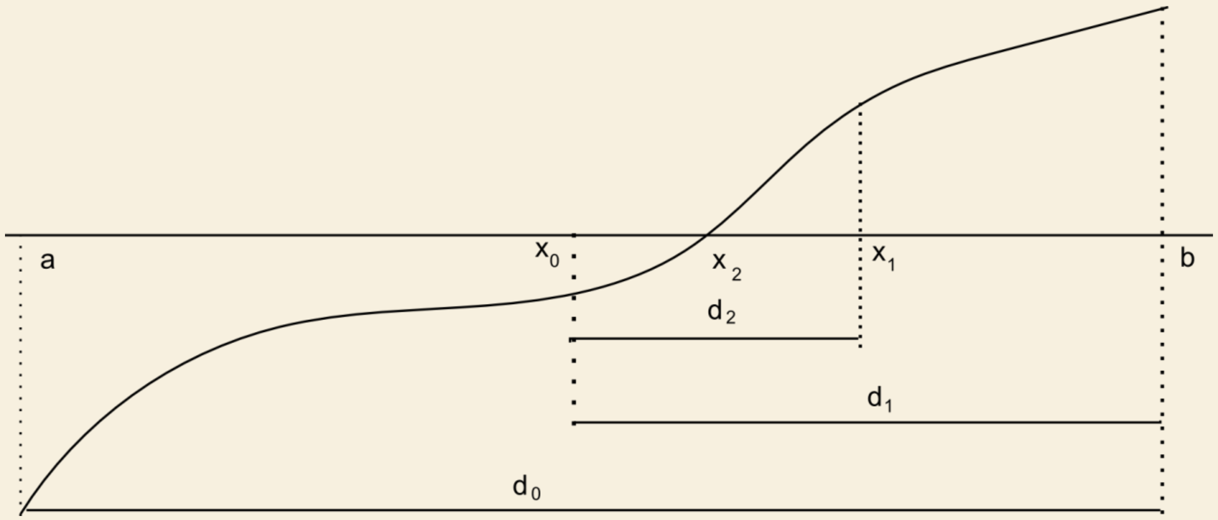
- Прямые методы
  - Конечное число элементарных операций
  - Точны в смысле отсутствия погрешности метода
  - Однако часто очень чувствительны к погрешности
  - Решение квадратных уравнений

$$x_{1,2} = -\frac{b \pm \sqrt{D}}{2a}$$

  - Метод Гаусса
  - …

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ

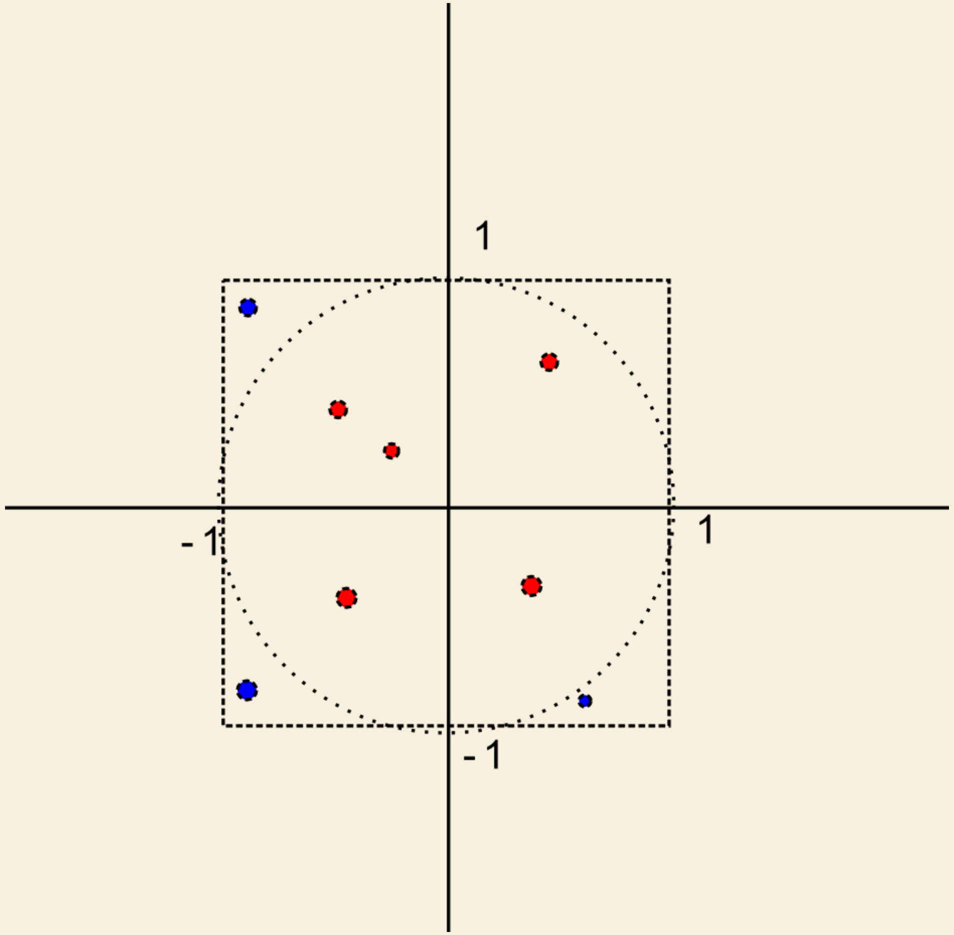
- Итерационные методы
  - Построение последовательных приближений
  - Однотипные набор действий
  - Точное решение получить невозможно
  - Априорная оценка погрешности
  - Не всегда сходятся



Например, метод половинного деления

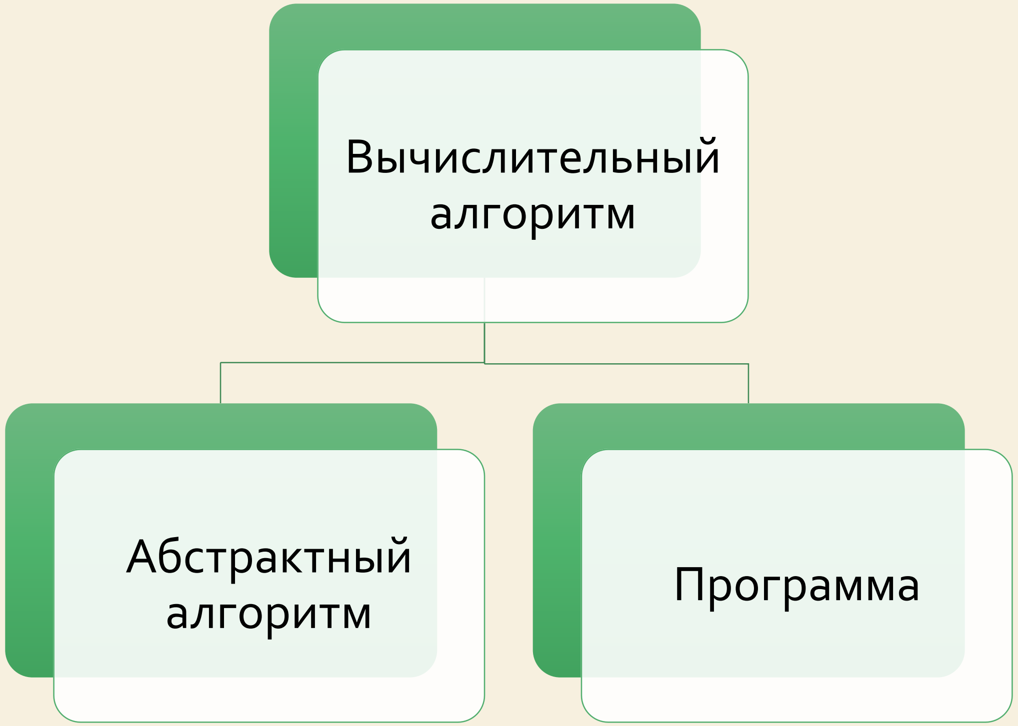
ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ

- Методы статистических испытаний
  - Поиск решения на основе построения статистики
  - Ресурсоёмки
  - Точное решение получить невозможно
  - Позволяют моделировать очень большие системы



ВЫЧИСЛИТЕЛЬНЫЙ АЛГОРИТМ

- Точное описание последовательности действий для преобразования входных данных в результат
- Корректен, если:
  - Реализуется за конечное число действий
  - Устойчив по входным данным
  - Вычислительно устойчив



ВЫЧИСЛИТЕЛЬНАЯ УСТОЙЧИВОСТЬ

Вычислим

$\int_0^1 x^n e^{1-x} dx$

для различных значений *n*

$$I_n = \int_0^1 x^n d(-e^{1-x}) = -x^n e^{1-x}\Big|_0^1 + \int_0^1 x^{n-1} e^{1-x} dx = nI_{n-1} - 1$$

$$I_0 = \int_0^1 e^{1-x}\, dx = e - 1 \approx 1.71828$$

ВЫЧИСЛИТЕЛЬНАЯ УСТОЙЧИВОСТЬ

- Кажется, всё хорошо, но

$$I_n = nI_{n-1} - 1 = n\big((n-1)I_{n-2} - 1\big) - 1 = \cdots$$

$$=> \ \Delta I_n \sim n!\,\Delta I_0$$

- Небольшая ошибка в вычислении растёт как факториал

ТРЕБОВАНИЯ К ВЫЧИСЛИТЕЛЬНЫМ АЛГОРИТМАМ

1. Экономичность
2. Точность
3. Экономия памяти
4. Простота

Понятие $O(\cdot)$

- Понятие, относящееся к скорости роста функций
  - $f(n), g(n), f(n) = O\big(g(n)\big)$ если $\exists c : f(n) \leq cg(n)$
- Например, $3n^2 + 10n - 4$ имеет порядок $O(n^2), n > 1$, потому, что

- $3n^2 + 10n - 4 \leq 4n^2$, в этом случае $c = 4$

При этом *c* может быть, вообще говоря, большим

ПОНЯТИЕ $\Omega(\cdot), \Theta(\cdot)$

- Аналогично
- $f(n), g(n), f(n) = \Omega(g(n))$ если $\exists c : f(n) \geq cg(n)$
- То есть $f(n) = \Omega(g(n)) \Leftrightarrow g(n) = O(f(n))$
- $\Theta(\cdot)$ - если одновременно $\Omega(\cdot)$ и $O(\cdot)$

- Мы будем использовать эти понятия в двух ситуациях:

1. Оценивая количество действий алгоритма, т.е. *экономичность*
2. Оценивая величину погрешности, т.е. *точность*

ЭКОНОМИЧНОСТЬ АЛГОРИТМОВ

- Число элементарных операций
- Как правило, говорят об порядке количества операций
- Правило Крамера

$$Ax = b$$

$$x_i = \frac{\Delta_i}{\Delta}$$

Для вычисления x_i нужно посчитать *(n+1)* определитель порядка ***n***

Общее количество вычислений *(n+1)!*

Для решения системы из 21 уравнения нужно 507 лет

ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ ВЫЧИСЛИТЕЛЬНЫХ АЛГОРИТМОВ

1. Экономичность
2. Точность
3. Экономия памяти
4. Простота
5. Надёжность
6. Работоспособность
7. Переносимость
8. Поддерживаемость

ЗАДАЧА

- Попробуйте применить метод половинного деления для отыскания минимума функции
- Какие ограничения есть у этого метода?
- Какой это метод – прямой или итерационный?
- Сможете ли вы оценить количество действий?

Для оценки работоспособности используйте функцию

$$f(x) = x^4 + 4x^3 + 6x^2 + 4x + 1, x \in [-2,2]$$